An Efficient Off-Line E-Cash System based on Signcryption without Bilinear Pairings

Hassan M. Elkamchouchi  
Alexandria  
Egypt

Eman F. Abou Elkheir  
Kafr Elsheikh  
Egypt

Yasmine Abouelseoud  
Alexandria  
Egypt

ABSTRACT
There is an increased activity in research and development conducted to improve current payment systems in parallel with the progress of Internet. Signcryption combines the functionalities of encryption and digital signing in a single logical step. It is a cryptographic primitive that provides confidentiality, integrity, authentication and non-repudiation. Identity-based cryptography serves as an efficient alternative to the traditional certificate-based cryptosystems. This paper introduces an efficient electronic payment system based on signcryption without bilinear pairings. This e-cash system is more efficient than other schemes employing bilinear pairings and involves less computational cost. In the proposed protocol, the token is issued and authenticated by the bank to prevent double spending problem. The customer delegates his signing capability to the merchant. The bank verifies the identities of both the original signer (customer) and the proxy signer (merchant) and ensures the originality of the transaction. Unlike the existing e-payment systems, the problem of double spending of e-cash does not arise because each transaction is made uniquely identifiable. Hence, no separate protocol is needed to check double-spending. The performance and the security analysis of the proposed e-cash system are discussed revealing its strength from the viewpoint of security and efficiency with regard to computations required.

General Terms
Cryptography, Security.

Keywords
E-cash payment system, Signcryption, Bilinear Pairings, E-Commerce Security

1. INTRODUCTION
In the last years, cryptographic protocols and network technologies have experienced rapid development [1, 2, 3, 4]. Electronic commerce is one of the most important applications over the Internet. Smart card based remote user authentication [5, 6] is the simplest and most convenient authentication mechanism for insecure networks. Customers’ privacy must be protected if they are involved in legal commercial transactions or payments. Blind signature schemes [7] have been widely used to protect the right of customer’s privacy in untraceable electronic cash systems [8]. However, it is easy to make multiple copies of the electronic coin, which is in the form of number strings. Therefore, anonymity revocation mechanisms are used in order to eliminate the possible abuse of unlinkability. If a user makes an abuse in the e-cash system; such as double spending, blackmailing or money laundering, then a trusted third party runs a specific protocol (tracing protocol) in order to reveal his/her identity. Chaum [8] proposed the first untraceable electronic cash system based on blind signatures in 1982. Several extensions have been proposed, which provide functionalities such as anonymity revocation and double spending prevention [9, 10, 11].

In the field of e-cash systems, the notions of on-line systems and off-line systems refer to a specific property of the payment protocol. On-line e-cash systems require constant and real time involvement of the bank in every payment transaction, resulting in excessive communication and computational costs. In contrast, off-line e-cash systems usually operate in dual mode; verifying high cost transactions on-line, while the rest of the payments are processed in batch mode by the bank.

The first off-line electronic cash system was introduced by Chaum in 1990 [12] and then further developed by Popescu in 2006 and 2009 [13,14], by De Santis in 2007 [15], by Chou in 2009 [16], and by Au in 2008 [17]. In these papers, the bank is not involved in the payment transaction between a customer and a merchant. Customers withdraw electronic coins from the bank and use them to pay a merchant (a shop). Then, the merchant deposits the coins back to the bank. Off-line e-cash systems use a trusted third party (TPP) to trace the criminals in order to protect the honest participants of the e-cash system.

Miller [18] and Koblitz [19] introduced Elliptic Curve Cryptography (ECC), which has increasingly attracted the attention of researchers in recent years due to its shorter key length requirement in comparison with other public key cryptosystems based on finite fields; such as DSA [20], ElGamal [21] and RSA [22]. For example, the 160-bit elliptic curve version of the DSA signature algorithm (ECDSA) has a security level equivalent to 1024-bit DSA signature algorithm. Such advantages make elliptic curve cryptography a better choice for public key cryptography. An E-cash transfer system using blind signatures based on the elliptic curve discrete logarithm problem has been recently proposed by Popescu in 2009 [13]. Another scheme employing elliptic curves is that proposed by Debasis Giri and Arpita Mazumdar in 2013 [23].

This paper introduces a new e-cash system employing identity-based signcryption without bilinear pairings. The proposed e-cash protocol is based on the signcryption scheme in [24] and the proxy signcryption scheme in [25] by which the customer delegates his signing rights to the merchant.
The rest of the paper is organized as follows. In the next section, the requirements for a secure e-cash system are summarized. The basic steps of the proposed e-cash system are presented in Section 3 and a more detailed description of the system appears in Section 4. The performance and security of the proposed system are analyzed in Section 5. Finally, Section 6 concludes the paper.

2. Security Requirements for any E-cash System
The most important requirements [14, 16] for a secure electronic cash transfer system are summarized below. They are:

2.1 Mutual authentication
Two parties can authenticate each other correctly.

2.2 Correctness
One can ensure the correctness and integrity of messages transmitted by the other designated party.

2.3 Unforgeability
Only the authorized bank can issue coins.

2.4 Traceability
The bank can reveal the identity of customer (with the trustee’s help) if the same e-coin is spent twice.

2.5 Efficiency
The e-cash system must be efficient in terms of storage requirements and computations.

3. The Proposed E-cash System
An e-cash system is a set of entities with their interactions, exchanging e-cash and goods. Our system has three entities:

Customer(C): purchases goods or acquires services from the merchant using the e-cash.

Merchant(M): sells goods or services to the customer, and deposits the e-cash to the bank.

Bank(B): issues the e-cash and manages the accounts for customers and merchants.

There are also three protocols in the system: withdrawal, payment and deposit. The customer withdraws a token from the bank and pays the token to the merchant. The token is readable only. The merchant gets the token from the customer and deposits it in the bank. The bank manages customer accounts, issues and updates tokens. No separate protocol is required to trace a dishonest customer. The proposed protocol is shown in Figure 1. In a transaction, the following events take place:

At first, setup is done by a central authority (CA).
1. (C sends to B) The customer requests the withdrawal of an electronic coin (token) from his account in the bank.
2. (B sends to C) Bank issues a signed token to C and securely sends it to C and deducts the appropriate amount from the customer’s account.
3. (C sends to M) C receives the token, chooses an item from M's home page and sends the signencrypted order information (OI) to M. C also sends the hash form of the token information signed which later helps B to verify C.
4. (M sends to C) M sends the hash value of SEQNO of the token signed as an acknowledgement to C. M also delivers the products to the customer. Transfer of products may be immediate (for intangible goods) or delayed (for tangible goods).
5. (C sends to M) The customer delegates his signing rights to the merchant and the bank acts as the verifier.
6. (M sends to B) Merchant appends price details, own account number (MAC) to OI and forwards the signencrypted (with proxy signature) modified OI to B along with the signed hashed token.
7. (Verification by B) Bank verifies both the (proxy signer) merchant as well as the (original signer) customer and ensures that OI is genuinely placed by the customer. The bank retrieves the customer token details by matching (TOKEN_ID||SEQNO). It also performs hashing and checks whether the sent hashed token value is identical to the hash value of the stored token.
8. (B sends to C) After verification, the bank sends update information to C via e-mail. Updated token information is kept in a server. A valid customer after a certain period of time can download the updated token details (Token value, SEQNO, TS).
9. (B sends to M) Bank credits the account of the merchant and sends an acknowledgement to it.

![Fig. 1The proposed e-cash protocol steps](image)

In the proposed protocol, the format of the token information (TI) is shown below.

<table>
<thead>
<tr>
<th>A/C NO</th>
<th>TOKEN_ID</th>
<th>SEQNO</th>
<th>EXPIRES</th>
<th>TS</th>
<th>VALUE</th>
<th>PROPOS</th>
</tr>
</thead>
</table>

A/C NO: Account number of customer
TOKEN_ID: Unique identification number of a token
SEQNO: Unique sequence number for each transaction
EXPIRES: Lifespan of a token.
TS: Time stamp of a transaction.
VALUE: Monetary value.

Moreover the format of Order Information (OI) is shown below.
where ITMCODE is a unique code of an item for sale.

4. The Detailed Description of the Proposed Protocol

4.1 Setup

Given a security parameter $k$, the key generator center (KGC) chooses $q$ a large prime number, where $q > 2^{160}$, (a, b) two integer elements which are smaller than $q$ and satisfy $(4a^3 + 27b^2) mod q \neq 0$. $E$ is the selected elliptic curve over a finite field $F_q : y^2 = (x^3 + ax + b) mod q$. $P$ is the base point of order $n$ of the group $G$ (the set of points on $E$ generated by $P$). Also, $O$ is the point at infinity and $n$ is the order of the point $P$, where $n$ is a prime, $nP = O$ and $n > 2^{160}$. The KGC selects a cryptographic one way hash function $H : \{0,1\}^l \times Z_q \rightarrow Z_q$.

4.2 Key Generation

The KGC selects a random number $s$ as the master key and computes the master public key $R = sP$. The KGC keeps $s$ secret and publishes the system parameters $\{k, G, P, R, H\}$. The KGC generates the secret and public keys of the customer, the merchant and the bank then sends the secret keys to their owners through a secure channel and publishes the public keys and the identities. The KGC calculates the customer, the merchant and the bank secret keys respectively as follows:

- $d_C = (h(ID_C)s) mod n$,
- $d_M = (h(ID_M)s) mod n$ and
- $d_B = (h(ID_B)s) mod n$.

The KGC calculates the public keys as follows: $Q_C = x_C, R$ (the customer public key), $Q_M = x_M, R$ (the merchant public key) and $Q_B = x_B, R$ (the bank public key).

4.3 Payment Protocol

Step 1: The customer $C$ sends a request for a token indicating the amount and his account number with the bank $B$ (token issuing authority). This information is sent securely by signcrypting the message, say $req \equiv (A/cno., Amt)$

The customer $C$ chooses a random number $W$ and computes:

- $k_C = ID_B, wQ_B = (\rho, v)$
- Split $\rho$ into $k_{1C}, k_{2C}$
- $c = E_{k_{1C}} (req)$
- $r = hash(req, k_{2C}, ID_C, ID_B)$
- $s = (ID_B, w - r, x_C, ID_C) mod n$
- $C$ sends $(c, r, s)$ to the bank.

Step 2: After receiving it, the bank verifies the authenticity of the received message. It performs the following steps.

The bank recovers the key $k_C$ as follows:

- $k_C = sQ_B + r.ID_C, x_B, Q_C = ID_B, wQ_B = (\rho, v)$
- Split $\rho$ into $k_{1C}, k_{2C}$
The customer chooses a random number $w_3$ and computes $r_3 = (w_3 + x_c) \cdot R = (u_1, v_1)$.

The merchant $M$ recovers the key $k_{2C}$ by computing

$$k_{2C} = s_2 \cdot Q_M + r_2 \cdot ID_C \cdot x_M \cdot Q_c = ID_M \cdot w_2 \cdot Q_M = (\rho_2, \omega_2)$$

Step 4:
The merchant checks the authenticity of the customer and retrieves the value of $O1$.

* The merchant $M$ retrieves the value of $O1$.

---

**The correctness of the equation is demonstrated below:**

$$RHS = \sigma \cdot R + ID_C \cdot h(\alpha, m_w) \cdot Q_C$$

$$= (ID_M \cdot d - x_c \cdot h(\alpha, m_w) \cdot ID_C) \cdot R + h(\alpha, m_w) \cdot ID_C \cdot Q_C$$

$$= ID_M \cdot d \cdot R - x_C \cdot h(\alpha, m_w) \cdot ID_C \cdot R + ID_C \cdot x_C \cdot h(\alpha, m_w) \cdot R$$

$$= ID_M \cdot d \cdot R = T = LHS$$

The bank recovers the key as follows:

$$\Delta R = r \cdot D_{2C} (c_2)$$

$$\tilde{r}_2 = \text{hash(OI, } k_{2C}, ID_C, ID_M)$$

* If $\tilde{r}_2 = r_2$, the merchant verifies that the $O1$ has been issued by a valid customer.

Also, the merchant authenticates that the signature has been issued by the customer as follows:

$h(TI) \cdot Q_c = (S_j - u_j) \cdot R$

After successful verification, $M$ sends the signature of $SEQNO$ (part of token information $O1$) to $C$ in the following manner:

* The merchant $M$ chooses a random number $w_4$ and computes $r_4 = (w_4 + x_M) \cdot R = (u_2, v_2)$.

**Step 5:** The customer $C$ verifies the merchant as follows:

$$SEQNO_{Q_M} = (S_j - u_2) \cdot R$$

Then, the customer delegates his signing rights to the merchant.

The original signer ($C$) chooses a random number $d \in [1, q - 1]$ and computes:

1. $T = ID_M \cdot d \cdot R = (\alpha, \beta)$

2. $\sigma = (d, ID_M - x_C \cdot h(\alpha, m_w, ID_C)) \cdot mod \ n$

C sends $\sigma$ to the proxy agent (merchant $M$).

**Step 6:** The proxy ($M$) authenticates the original signer (customer) as follows:

$$\sigma, \tau + ID_C \cdot h(\alpha, m_w) \cdot Q_C = T$$

...
The bank retrieves the customer's token details by matching (TOKEN ID||SEQNO). Also, it performs hashing and checks whether the hashed token value sent by customer is the same as the hash value of the stored token.

Step 8: After verification, the bank sends update information to C via e-mail. Updated token information is kept in a server. A valid customer after a certain period of time can download the updated token details: Token value, SEQNO, TS.

Step 9: The bank credits the account of merchant and sends an acknowledgement.

5. Security and Performance Analysis

The security of the proposed e-cash protocol is based on the elliptic curve discrete logarithm problem (ECDLP) [26]. Up till now, the ECDLP is considered to be hard.

The Elliptic Curve Discrete Logarithm Problem (ECDLP) is defined as follows. Let $G$ and $Q$ be two points on an elliptic curve and $G$ is of order $n$ and $n$ is a prime. The point $Q = kG$, where $k < n$. Given these two points $G$ and $Q$, find the discrete logarithm of $Q$ to the base $G$; that is, $k$.

5.1 Security Analysis

5.1.1 Mutual Authentication

The proposed protocol uses the signcryption primitive in which the message is efficiently signed and encrypted in a single logical step, so the communicating parties can authenticate each other easily. For example, in the withdrawal phase between the customer and the bank; the customer sends a signcrypted request $(r, c, s)$ to the bank for the withdrawal of electronic coins from his account. If an adversary wants to masquerade as the customer to send an encrypted message to the bank, the bank will reject it since it will recover a wrong key and it cannot decrypt the ciphertext to obtain any meaningful information and surely authentication fails. Moreover, only the bank can recover the sent message as it is the only entity in possession of the secret key required.

5.1.2 Unforgeability

The unforgeability property holds in all exchanged messages due to the employment of secure (proxy) signature and (proxy) signcryption schemes [24,25].

5.1.3 Traceability:

In the proposed protocol, the bank verifies the original signer (C) and the proxy signer (M) and checks the validity of token. A token cannot be double spent as it is updated. In other e-payment protocols, detection of double spending involves a database search and the tracing phase involves the bank and a trusted third party in order to detect the identity of a double spender.

5.2 Comparative study

Table I shows the time abbreviations that will be used in the comparison table. Table II shows a comparison between the performance of the proposed protocol to that of the protocol in [23]. The comparison in table II shows that the proposed protocol is more efficient and has a lower computational cost than the protocol in [23] with bilinear pairings. Also, using the proposed protocol saves bandwidth, where the signcryption module involved in the proposed protocol reduces signaling overhead.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{EC-mult}$</td>
<td>time required for executing multiplication operation on elliptic curve $E$</td>
</tr>
<tr>
<td>$T_{EC-add}$</td>
<td>time required for executing addition operation on elliptic curve $E$</td>
</tr>
<tr>
<td>$T_{mult}$</td>
<td>time required for executing modulus multiplication in a finite field</td>
</tr>
<tr>
<td>$T_h$</td>
<td>time required for executing one way dispersed row function operation</td>
</tr>
<tr>
<td>$T_{enc}$</td>
<td>time required by the system for executing encryption operation</td>
</tr>
<tr>
<td>$T_{dec}$</td>
<td>time required by the system for executing decryption operation</td>
</tr>
<tr>
<td>$T_{exp}$</td>
<td>time required for executing modulus exponential operation</td>
</tr>
<tr>
<td>$T_{pairings}$</td>
<td>time of executing a bilinear pairing operation</td>
</tr>
</tbody>
</table>

6. Conclusion

In this paper, a new e-cash payment protocol based on signcryption has been introduced. It involves no bilinear pairings. The proposed protocol is presented together with its security analysis. It is also compared with another e-cash protocol in [23]. The comparative study shows that the proposed protocol is more efficient reducing the signaling overhead and the computational cost while achieving the same security properties of the protocol in [23].
Table 2 the comparison between the proposed protocol and the protocol in [23]

<table>
<thead>
<tr>
<th>Step</th>
<th>The protocol in [23](with bilinear pairings)</th>
<th>The proposed protocol(without bilinear pairings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(Signcryption)</td>
<td>$3T_{EC}$-mult + $2T_h$ + $3T_{pairings}$ + $1T_{EC}$-enc + $1T_{dec}$</td>
<td>$1T_{EC}$-mult + $1T_h$ + $4T_{mult}$ + $1T_{EC}$-enc + $1T_{dec}$</td>
</tr>
<tr>
<td>2( verifying Signature +signcryption)</td>
<td>$3T_{EC}$-mult + $4T_h$ + $6T_{pairings}$ + $1T_{EC}$-add + $1T_{EC}$-enc + $1T_{dec}$</td>
<td>$3T_{EC}$-mult + $2T_h$ + $6T_{mult}$ + $1T_{EC}$-add + $1T_{EC}$-enc + $1T_{dec}$</td>
</tr>
<tr>
<td>3( verifying Signcryption +signcryption +signature)</td>
<td>$6T_{EC}$-mult + $6T_{pairings}$ + $2T_{EC}$-add + $1T_{EC}$-enc + $1T_{dec}$</td>
<td>$4T_{EC}$-mult + $3T_h$ + $7T_{mult}$ + $1T_{EC}$-add + $1T_{EC}$-enc + $1T_{dec}$</td>
</tr>
<tr>
<td>4( verifying signature + verifying Signcryption +signature)</td>
<td>$3T_{EC}$-mult + $4T_h$ + $7T_{pairings}$ + $1T_{EC}$-add + $1T_{dec}$</td>
<td>$5T_{EC}$-mult + $3T_h$ + $3T_{mult}$ + $1T_{EC}$-add + $1T_{dec}$</td>
</tr>
<tr>
<td>5( verifying signature + delegation signing rights)</td>
<td>$2T_{EC}$-mult + $2T_h$ + $3T_{pairings}$ + $1T_{EC}$-add</td>
<td>$3T_{EC}$-mult + $1T_{EC}$-enc + $3T_{mult}$</td>
</tr>
<tr>
<td>6( verifying signature + proxy key + proxy Signcryption)</td>
<td>$4T_{EC}$-mult + $3T_h$ + $4T_{pairings}$ + $3T_{EC}$-add + $1T_{Enc}$</td>
<td>$3T_{EC}$-mult + $1T_{h}$ + $5T_{mult}$ + $1T_{Enc}$</td>
</tr>
<tr>
<td>7( verifying signature + verifying proxy Signcryption)</td>
<td>$4T_{h}$ + $5T_{pairings}$ + $1T_{dec}$</td>
<td>$5T_{EC}$-mult + $3T_{h}$ + $3T_{mult}$ + $3T_{EC}$-add + $1T_{dec}$</td>
</tr>
<tr>
<td>Total</td>
<td>$21T_{EC}$-mult + $25T_{h}$ + $36T_{pairings}$ + $9T_{EC}$-add + $4T_{Enc}$ + $4T_{dec}$</td>
<td>$24T_{EC}$-mult + $14T_{h}$ + $31T_{mult}$ + $6T_{EC}$-add + $4T_{Enc}$ + $4T_{dec}$</td>
</tr>
</tbody>
</table>

7. REFERENCES


